Preprint of: **A. Reinhardt** and S. Koessler. "PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations." In: *Proceedings of the 9th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 2014, pp. 531–538.

PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations

Andreas Reinhardt, Sebastian Koessler

School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia
andreasr@cse.unsw.edu.au, sebastiank@cse.unsw.edu.au

***Abstract*—Distributed power meters (also termed *smart plugs*) are embedded systems that measure the electric power consumption of individual appliances at a fine temporal resolution. They enable a wide range of novel smart services, e.g., accurately forecasting power consumption or making recommendations how to save energy. However, distributed power metering combines high sampling rates with a potentially large number of monitored outlets. A torrent of power readings may thus be generated, incurring high bandwidth requirements for their transmission and a significant computational power demand for their processing. In this paper, we present a concept for the efficient local storage and processing of power consumption data called PowerSAX. Instead of operating on raw sensor readings, PowerSAX converts consumption data into their *symbolic representations* and thus mitigates their storage requirement. Subsequently, it enables embedded systems to recognize relevant patterns (*motifs*) in the symbolic representations of collected data. By only transmitting a message when a known motif is encountered in the sensor data, PowerSAX can significantly reduce an application's bandwidth requirements. We evaluate PowerSAX using real-world power consumption data and show to which extent smart plugs can make predictions of an appliance's future power demand.**

I. Introduction

The evolution of home automation technology has given rise to numerous novel smart building services. These services combine data from multimodal sources like wireless sensors, smartphones, and power meters. Based on the fusion of the collected data, smart home automation systems may control a building by means of actuator devices or make recommendations to the users, e.g., how to save energy. The potentially large volume of building sensor data required to realize these services, however, poses a new technological challenge for home automation systems. Phenomena with high dynamics, such as electric power consumption, require short response times and thus need to be sampled frequently, incurring a high bandwidth requirement for their data transmissions.

Numerous means for data compression have consequently been explored on embedded sensing systems (e.g., [1]) to reduce this demand for bandwidth while allowing for the reconstruction of the original data. In many cases, however, the detection of specific patterns in the data (e.g., indicators for an electric appliance's mode of operation) is much more relevant than the device's actual consumption data. As a result, the use of *events* was proposed in literature (e.g., [2], [3]). Events are defined before a sensor network is deployed. During runtime, all participating systems locally evaluate their readings and only transmit a message when all preconditions for an event are fulfilled. However, based on our prior observations in [4], the local detection of events in incoming sensor data streams often incurs a high demand for computational power.

We thus address these challenges by presenting PowerSAX, a novel approach for the lightweight representation of sensor data and the recognition of patterns therein. It achieves size reductions by converting sensing consumption values into their *symbolic representations*. In other words, it turns a time series of raw data points into a sequence of symbols. In contrast to existing work in this domain (e.g., [5]), however, the actual mapping between raw data and the symbolic representations is determined by a clustering algorithm and based on the histogram of the collected data. Besides creating near-optimal mappings for each appliance type, this approach also lowers the system's susceptibility to measurement noise. Characteristic patterns (*motifs*) are then identified in the symbol sequences for their later recognition in live data streams. Again, symbolic representations prove beneficial because they reduce the need for computation to recognize motifs; in fact, even embedded systems can recognize motifs composed of symbol sequences locally. Transmitting symbols instead of raw data points already leads to smaller packet sizes, and even higher savings can be achieved when only the identifiers of recognized motifs are transmitted instead of symbol streams. PowerSAX can thus measurably reduce the data volume while retaining all relevant features to enable smart home automation.

First, we summarize work related to symbolic approximations and pattern recognition in power time series in Sec. II. Next, we highlight the fundamental concept behind PowerSAX in Sec. III. We show how the symbol alphabets are defined based on historic sensor data in Sec. IV, and describe how characteristic motifs are extracted and subsequently recognized in real-time input data streams in Sec. V. We evaluate the accuracy of our motif detection system using real-world input data in Sec. VI and conclude this paper in Sec. VII.

II. Related Work

The realization of smart home services is generally based on the analysis of sensor data time series and their correlations. For example, identifying recurring patterns allows such systems to determine the user's habits and preferences. Similarly, deviations from regular patterns can indicate situations in which a home automation system's action is required to maintain the user's comfort and safety. Suitable means for the efficient analysis of time series are thus needed.

Preprint of: **A. Reinhardt** and S. Koessler. "PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations." In: *Proceedings of the 9th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 2014, pp. 531–538.

Transforming time series data into the frequency domain is a viable approach to lower the computational efforts required for their analysis, and widely used across many application domains, e.g., astronomy [6]. It reduces the computationally intensive analysis of time series data to the determination of the signal's spectral components. However, information about the temporal sequence of events is lost in this process. This effectively renders it inapplicable for smart home automation systems, where the order of events plays a major role.

In order to retain the temporal sequence of events in time series data and reduce the computational complexity at the same time, the concept of symbolic approximations (*SAX*) has been introduced in [5]. By converting raw data to symbolic representations prior to their analysis, the computational overhead for motif and discord detection (i.e., the identification of recurring patterns and outliers) is significantly lower than for raw time series. Symbolic representations inherently allow to suppress noise on signals and can even be used to detect scaling in both signal amplitude and duration (*dynamic time warping*). As symbolic approximations hence allow for the signal analysis in the temporal domain and alleviate the bandwidth requirement of sampled data points, they have been selected as the foundation for the work presented in this paper.

The extraction of higher-level information from smart power meter data has been presented in several papers. Initially started as Nonintrusive Appliance Load Monitoring (NALM) by George Hart [7], many further ideas to identify the presence and activity of appliances in buildings have been published (e.g., [8], [9], [10], [11]). While all of the approaches rely on the analysis of consumption data, however, temporal dependencies of more than a couple of seconds are generally not taken into account. Instead, the solutions are based on momentary measurements and at most short-term time series of the transients during appliance (de)activation.

In a preliminary experiment, we have extracted patterns from raw power consumption time series in order to make predictions of an appliance's deactivation time [4]. While confirming the general viability and potential of the approach, we have observed high computational efforts, even for small experiments. These are likely to hamper an implementation at large scale and have motivated us to explore the applicability of symbolic representations which have only been considered in the following two other works to the best of our knowledge. Wijaya et al. use symbolic approximations in order to make load predictions based on patterns in aggregate consumption data of complete households [12]. However, a log-normal distribution is being used to approximate the power consumption data characteristics. Thus, the approach can be expected to be suboptimal when consumption characteristics do not fit the chosen unimodal distribution. Similarly, Bondu et al. list the smart grid as a potential application domain for their OSAX approach, but do not consider such data in their evaluation [13]. PowerSAX overcomes these limitations by adapting to the actual distribution of the underlying power consumption data. Thus, it allows for higher precision when operating on time series that follow multimodal symbol distributions.

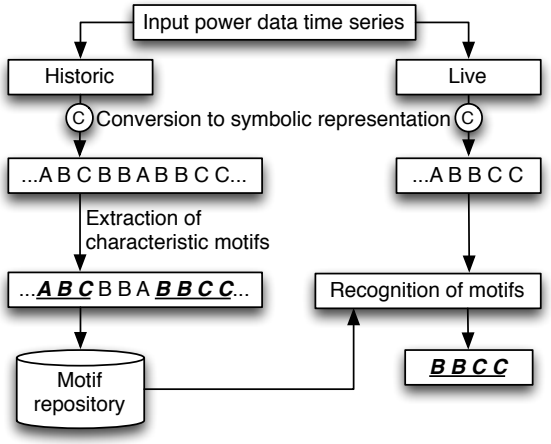


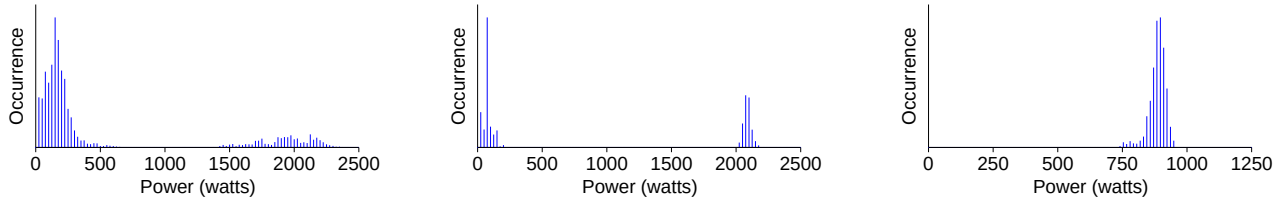
Fig. 1. PowerSAX system overview.

III. The PowerSAX Concept

The core idea behind PowerSAX is to use symbolic representations of power meter data in order to facilitate the efficient discovery and recognition of recurring patterns. We visualize its general data processing sequence in Fig. 1.

1) The input sequence is converted into its symbolic representation. This step is fundamental to both SAX and PowerSAX and is the simplification that speeds up any subsequent processing. In essence, the range of potential values in the input sequence is divided into a number of intervals, and each interval is allocated a unique symbol in the symbol alphabet. More details about this process step as well as its limitations when applied to power sensor data are explained in Sec. IV.
2) Characteristic motifs are extracted from historic data and stored in a repository. When analyzing electrical loads, these motifs often model the change of an appliance's operational mode or any characteristic sequences of power consumptions exhibited during the appliance's operation. Due to the translation of input sequences into their symbolic representations, these motifs are represented as sequences of symbols, as shown in Fig. 1. The criterion for adding a motif to the repository in this step is its recurring nature; PowerSAX only considers motifs that occur at least twice in the input data set. More details on the actual recognition of motifs and the allowed error thresholds are presented in Sec. V.
3) Live input data are converted to their symbolic representations in order to determine if they match any of the motifs in the repository. To this end, a window containing the most recent symbols is compared to all motifs in the repository. Once a previously stored motif is recognized, additional functionalities like load forecasts or the detection of abnormal appliance behavior become possible. We evaluate the potential of symbolic approximations to realize one such functionality in Sec. VI.

Preprint of: **A. Reinhardt** and S. Koessler. "PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations." In: *Proceedings of the 9th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 2014, pp. 531–538.



(a) Histogram of a washing machine. (b) Histogram of a dishwasher. (c) Histogram of a cooking stove.

Fig. 2. Power consumption histograms for different appliance types. Consumption values below 2 W (e.g., standby power) have been omitted for visual clarity.

IV. Symbol Alphabet Definition

Traditionally, the input data used in conjunction with symbolic approximations has been assumed to follow a Gaussian distribution (cf. [14]). Under this assumption, the conversion of raw data to their symbolic form works as follows. At first, a z-normalization step is applied to the signal to make it zero-mean and of unit variance [15]. Subsequently, the area under the probability density function curve is divided into segments of equal area, each of which represents the mapping from an input data range to its corresponding symbol.

However, a major limitation arises for the use of this approach with distributed power data. Firstly, the normalization to unit variance effectively removes the absolute dimension of the input data. As a result, symbolic representations lose all information about the amplitude of the original signal. However, electrical appliances can be expected to exhibit highly characteristic power consumption levels throughout all operation cycles in their lifetime. To retain the amplitude information of power measurements, PowerSAX hence omits the z-normalization step. Secondly, the assumption of a Gaussian symbol distribution no longer holds true for data that has not undergone z-normalization. We have visualized the symbol distribution histograms of three household appliances from the Tracebase project [16] in Fig. 2. With the exception of the electric stove depicted in Fig. 2c, all histograms show several power consumption peaks. As power consumption values around zero occur for all devices (but were omitted from the diagrams for the sake of clarity), each appliance type has at least two distinct power consumption values. In other words, all analyzed power consumption data follow multimodal distributions and thus cannot be modeled as unimodal Gaussian distributions without introducing significant errors.

A. Alphabet Definition in PowerSAX

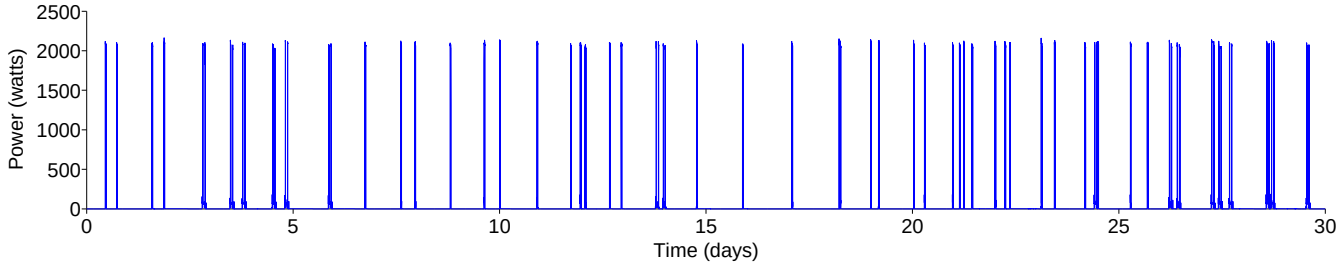
Having shown that unimodal distributions are unsuited to reflect the characteristics of distributed power meter data properly (cf. Fig. 2), an alternative solution needs to be found. Let us reconsider the histogram of the dishwasher shown in Fig. 2b. It shows two distinct consumption peaks. While values around 100 W can be attributed to the rinsing phases, consumptions above 2,100 W originate from the activity of the heater elements. A third peak of 0 W amplitude (not shown) exists which reflects phases during which the appliance was turned off.

Intuitively, the optimal number of symbols thus appears to be dependent on the number of distinct clusters in the power consumption histogram. Allocating too few symbols for the alphabet would result in two distinct consumption clusters to be merged, thus losing the ability to distinguish between the appliance's heating and rinsing phases. Similarly, using too many clusters would potentially lead to situations in which two similar consumption values are being associated to different symbols, although they relate to the same operational mode.

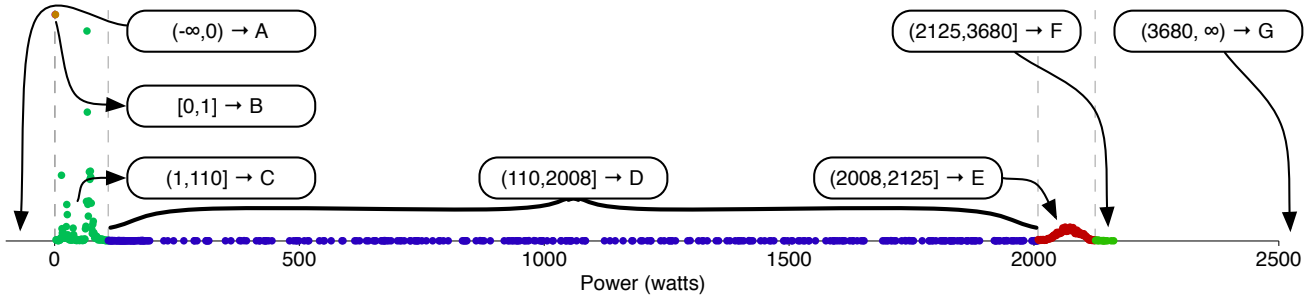
Defining the number of used symbols in traditional SAX mainly serves to optimize the trade-off between speed and accuracy. As highlighted above, however, an appliance's power consumption histogram often reveals the distinct power consumption clusters that relate to the most frequently occurring operational modes. We hence propose to use a clustering-based approach to extract the symbol alphabet from historic data, which works as follows.

1) We use historic power consumption data from the appliance, segment them into non-overlapping windows of a constant size $\frac{w}{s}$ (cf. Sec. V for details), and calculate the mean value for each window. Subsequently, the histogram of all resulting mean values is computed.
2) The histogram is clustered by applying the MeanShift algorithm [17]. It annotates each data point in the histogram by its membership to a cluster. Power consumption clusters are thus implicitly created, with their boundaries represented by the extremal members of each determined cluster. MeanShift has been chosen as it can determine the optimal number of required clusters itself.
3) Once all clusters have been detected, PowerSAX analyzes whether all possible input values are covered by the alphabet. If gaps exist between two neighboring cluster boundaries, additional intervals are added to the symbol alphabet. Despite the fact that these values have never or very rarely occurred in the past, we define them in the model in case they occur in data measured later.
4) PowerSAX finally adds intervals for $(-\infty, 0)$ and (P_{max}, ∞) that capture erroneous readings. Herein, P_{max} is the maximum output value of the sensors. As a result, the resulting alphabet is capable of mapping the entire range of potential input values to their symbols.
5) Finally, symbolic names are attributed to the extracted clusters by consecutively labeling them and storing them in the form of a look-up table.

Preprint of: **A. Reinhardt** and S. Koessler. "PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations." In: *Proceedings of the 9th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 2014, pp. 531–538.



(a) Actual input data trace of a dishwasher's power consumption over a sampling period of 30 days.



(b) Histogram of the power consumption values with extracted symbol clusters and corresponding symbol annotations.

Fig. 3. Output mapping based on histogram clustering by means of the MeanShift algorithm.

This process is visualized in Fig. 3 for a 30-day power consumption trace from a dishwasher. The actual power consumption is shown in Fig. 3a and alternates between phases of activity (with peaks around 100 W and 2,100 W) and inactivity (0 W). The corresponding histogram entries are shown as dots in Fig. 3b, which also shows the resulting clusters and their corresponding symbols. Clusters boundaries are indicated by dashed vertical lines, and the gaps between any two clusters determined by the MeanShift algorithm (e.g., all values between 110 W and 2,008 W) were covered by adding a new cluster D (cf. step 3). The measurement limit P_{max} of the hardware power sensors was 3,680 W. Hence, clusters A and G are exclusively comprised of invalid values, and such symbols will be disregarded in subsequent processing steps in order to avoid the consideration of erroneous data.

V. Motif Recognition

After having converted the time series data to their symbolic representations, recurring sequences and their temporal dependencies can be determined. We use the notion of *motifs* as defined by Lin et al. as "previously unknown, frequently occurring patterns" [18], and describe the process of their extraction from historically observed data as follows.

A. Windowing

Sensor data are commonly sampled at a fixed rate (e.g., one reading per second). As motifs are designed to represent temporal dependencies of the input data, however, they need to model the activity within an analysis time window. PowerSAX uses a sliding sampling window of length w, upon which all further data processing operations are based. The window size w needs to be defined depending on application and performance constraints. Exactly w input readings are required to decide if the real-time input data matches a known motif. While low delays can be achieved by selecting small window sizes, motifs that exceed the window size cannot be captured completely and in consequence not be detected easily, if at all. In contrast, large window sizes may help to extract more relevant motifs, but come at the cost of larger detection delays.

B. Motif Extraction

Let us assume that the analysis window size w has been defined based on the characteristics of the raw input data and the actual application requirements. In order to allow for the extraction of motifs, the raw input data in each of the windows must be converted to their symbolic representations first. This process is performed based on the look-up table extracted according to Sec. IV. However, the time window is first segmented into s sub-windows of equal size (each comprised of $\frac{w}{s}$ raw data points, as mentioned in Sec. IV-A). Besides eliminating further measurement noise, sub-windowing also reduces the number of symbols in each motif and thus increases the applicability of PowerSAX on embedded systems.

All resulting motifs are exactly s symbols long, which we call their *symbolic length*. Naturally, the number of resulting symbols s in each window w cannot be larger than the number of input data points (i.e., $s \leq w$). Only one output symbol is being generated for each of the sub-windows by converting the mean value of all samples in the sub-window to its symbolic representation. E.g., to convert $w = 128$ entries into a sequence with symbolic length of $s = 16$, at first w is divided into s sub-windows, comprised of $\frac{w}{s} = 8$ raw data points each. Then, the mean values of the 8 values in each sub-window are computed, and the corresponding symbolic representations returned.

Preprint of: **A. Reinhardt** and S. Koessler. "PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations." In: *Proceedings of the 9th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 2014, pp. 531–538.

In the training phase, PowerSAX extracts all candidate motifs (i.e., series of s symbols) from the historic data by means of sliding a window of size w across the entire input sequence. Each of the resulting symbolic strings is added to a list of candidate motifs, and annotated by the number of its occurrences. Keeping track of each candidate's number of occurrences is an integral part of PowerSAX in order to later determine whether a sequence fulfills the motif criterion (i.e., it occurs sufficiently frequently). Due to the applied averaging and quantization, the same symbolic string often occurs in two or more adjacent time windows. However, as the actual data only contain a single occurrence of the underlying motif, we merge all its subsequently detected occurrences into a single candidate entry without increasing its occurrence counter.

Upon completion of the motif extraction, the algorithm outputs a list of tuples, which store the candidate motifs as well as their number of unique occurrences. From this list, the features to be used in the real-time recognition phase can be selected based on the required characteristics. An example selection criterion for electrical appliances is the number of a motif's occurrences, as frequently occurring motifs often relate to changes in the underlying appliance's mode of operation. In the implementation described in this paper, PowerSAX thus adds all motifs with more than one disjunct occurrence to its motif repository, i.e., the list of motifs that can be matched later. The minimum occurrence requirement of motifs can, however, be easily adjusted to meet the application's needs.

C. Motif Recognition

The static definition of motif lengths allows for an efficient matching between the last s symbols of incoming stream data and the motifs stored in the repository. To this end, the most recent w values of streamed input data are converted to their symbolic representation and used as a query to the motif repository. As both sequences have the same length, determining their similarity can be easily done by means of computing the distance between each of the symbol pairs according to a distance metric.

With the symbol alphabet being autonomously extracted from the histogram, situations might occur in which very small clusters exist. As a result, motifs may not exactly match the real-time input data exactly, but show small discrepancies, e.g., deviate by just a single symbol. To encounter this case, we have designed PowerSAX to not only consider exact matches, but allow for a configurable error margin. For this purpose, we define the distance between two symbols c (in the candidate motif) and q (in the query) according to Eq. (1).

$$d(c, q) = |\beta_c - \beta_q| \tag{1}$$

We use the value of each symbol's upper cluster boundary β_c (i.e., the highest power consumption value of this cluster) as its nominated weight. Only for the first cluster, which spans all values from negative infinity to zero, this value is assumed to be $-\infty$ to allow for easier processing of erroneous readings. The resulting distance matrix for the example made in Fig. 3b is shown in Table I. This distance matrix is subsequently used in combination with the Euclidean distance calculation (cf. Eq. (2)) when determining the distance between the candidate motif $\hat{C}$ and query motif $\hat{Q}$, both of which are PowerSAX sequences of s symbols length.

TABLE I
DISTANCE MATRIX FOR DATA FROM FIG. 3.

	A	B	C	D	E	F	G
A	∞	∞	∞	∞	∞	∞	∞
B	∞	0	109	2007	2124	3679	∞
C	∞	109	0	1898	2015	3570	∞
D	∞	2007	1898	0	117	1672	∞
E	∞	2124	2015	117	0	1555	∞
F	∞	3679	3570	1672	1555	0	∞
G	∞	∞	∞	∞	∞	∞	∞

$$dist(\hat{C}, \hat{Q}) = \sqrt{\frac{w}{s} \times \sum_{i=1}^{s} d(c_i, q_i)^2} \tag{2}$$

Unless the symbolic representation of the input sequence contains the invalid value of ∞, PowerSAX thus queries the motif database by calculating the distance between the input sequence and every stored motif. Any resulting value below a user-definable threshold distance will be reported as a match.

VI. Evaluation

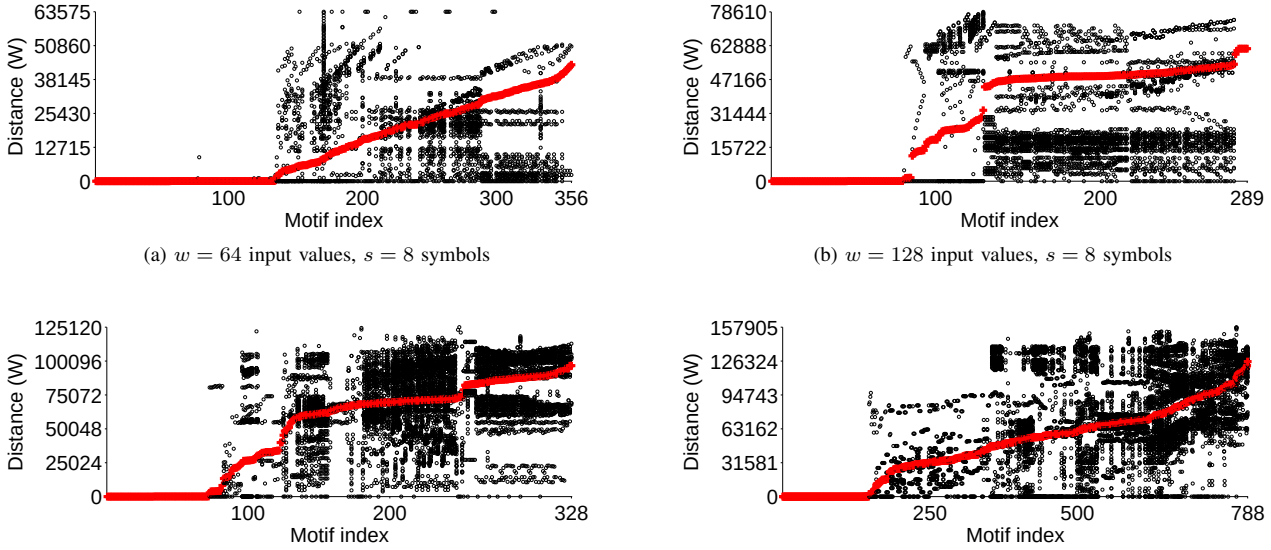
After having described the operation of PowerSAX, we assess its benefits for a potential use case of future smart homes, namely appliance power consumption prediction. We analyze the impact of the parameter selections on the prediction accuracy of PowerSAX first, and subsequently assess its potential for operation on embedded sensing systems.

A. Test Case: Power Consumption Prediction

By enabling smart plugs to make predictions for the power consumption of an attached load, they can help utility companies predict power peaks and thus help them to avert blackouts. To achieve this functionality, we use PowerSAX with a minor modification. In addition to the extraction of recurring motifs from the training data set, we annotate each of them by the $4 \times s$ symbols observed after their occurrence. Thus, we ensure that each recognized motif allows PowerSAX to make a consumption forecast, which is subsequently reported.

To cater for the realistic nature of the evaluation, we have used device-level power consumption data from the Tracebase project [16]. Thirty consecutive days of a dishwasher's power consumption (ID *B82F81*) were used, with sixteen of them for the training phase, and fourteen traces to test the system's prediction performance. Again, PowerSAX has been configured to add all motifs with at least two occurrences in the training data set to the repository. Subsequently, a sliding window of s symbols length is moved along the *testing* time series. The distance between all saved motifs and the sequence in the sliding window ($\hat{Q}$) is calculated according to Eq. (2), and the closest match that fulfills $dist(\hat{C}, \hat{Q}) < 2\,\mathrm{W}$ is reported. Note that we have particularly chosen a threshold value slightly

Preprint of: **A. Reinhardt** and S. Koessler. "PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations." In: *Proceedings of the 9th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 2014, pp. 531–538.



(a) $w = 64$ input values, $s = 8$ symbols

(b) $w = 128$ input values, $s = 8$ symbols

(c) $w = 256$ input values, $s = 8$ symbols

(d) $w = 512$ input values, $s = 8$ symbols

Fig. 4. Comparison of prediction errors for four different window sizes.

greater than zero in order to minimize the impact of measurement noise introduced by the power sensor hardware. Once a motif in the repository is recognized during the system's operation, its annotation is returned as the estimated future power consumption of the testing time series.

In order to evaluate the accuracy of a prediction, we also use the aggregate distance measure $dist$. More precisely, we compute the distance between the actual power consumption (i.e., the subsequently recorded data points of the testing time series) and the motif's annotated prediction. As highlighted above, the distance is calculated across the complete prediction window of $4 \times s$ symbols length.

B. Impact of Window Size w and Symbolic Length s

Let us start our evaluation with an analysis of the impact of the window size w. As highlighted in Sec. V-A, the window size determines the number of input samples required before a prediction can be made. In case of predicting an appliance's power consumption, however, the window size also determines the duration for which a prediction can be emitted. We use window sizes of 64, 128, 256, and 512 seconds to evaluate the error between the actual consumption observed in our testing data and the predictions emitted by PowerSAX.

The resulting distances between prediction and actual consumption for $s = 8$ are visualized in Fig. 4. In the subfigures, the x-axis lists all motifs that have been extracted from the training data set. For each of them, black dots visualize the distance between its made predictions and the actual testing data following the position where the motif has matched. The average prediction error for each motif is furthermore shown in red. We have sorted the x-axis by the observed average prediction errors for better visual clarity. The interpretation of the figures is straightforward. For example, Fig. 4a conveys that a total number of 356 motifs were extracted from the training data. Of these, 134 led to predictions with small average errors (below $60\,W$), whereas the remaining 222 motifs led to predictions with significantly larger average errors. All four figures confirm that some motifs have very good prediction capabilities across all analyzed parameter variations.

Across all regarded window sizes, the number of motifs that lead to predictions below the $60\,W$ error threshold varies (134 for $w = 64$, 80 for $w = 128$, 72 for $w = 256$, 146 for $w = 512$; values for other symbolic lengths in Table II). However, while a large number of motifs in the repository may increase the likeliness that one of them is recognized in the testing data, a much more important aspect is the overall accuracy of their predictions. Returning an erroneous prediction is less favourable than not emitting a prediction at all. We have thus determined the overall fraction of motif matches that lead to correct forecasts in Fig. 5.

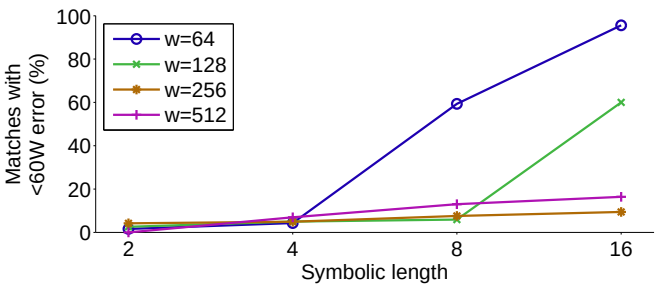


Fig. 5. Overall percentage of correct predictions when up to $60\,W$ error between prediction and actual data are allowed.

Preprint of: **A. Reinhardt** and S. Koessler. "PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations." In: *Proceedings of the 9th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 2014, pp. 531–538.

TABLE II
COMPARISON OF NUMBER OF MOTIFS WITH PREDICTION ERRORS BELOW 60 W AND TOTAL NUMBER OF EXTRACTED MOTIFS (IN BRACKETS).

Symbolic length	Window size w=64	w=128	w=256	w=512
s=2	3 (53)	6 (73)	16 (92)	9 (128)
s=4	28 (133)	25 (126)	36 (194)	60 (434)
s=8	134 (356)	80 (289)	72 (328)	146 (788)
s=16	5827 (6162)	528 (1003)	176 (683)	257 (1188)

From the figure, it can be observed that the motifs extracted when using smaller window sizes succeed much better in making correct predictions. In fact, up to 96% of its predictions are correct when applying the combination of long symbolic lengths and short windows. This can be primarily explained by the fact that smaller matching windows also lead to smaller prediction horizons, which are easier to predict accurately. Moreover, the distance definition in Eq. (2), which we have adopted from SAX [5], includes a dependency on the window size and thus implicitly penalizes prediction errors made when longer windows are being used. However, we have also observed that the averaging effect of larger window sizes in conjunction with shorter symbolic lengths creates more confusion between similar motifs. For these parameter ranges, PowerSAX is no longer able to discriminate between similar motifs with different predictions, resulting in a large percentage of incorrect forecasts. To sum up our general observations, the results allow us to conclude that motifs composed of more symbols are better suited to correctly predict the future consumption, and this observation is independent of the used window size.

Increasing the symbolic length hence appears as a promising means to cater for more correct predictions. However, practical consideration of the memory size restrictions of the embedded systems in smart plugs may hamper the use of long motifs. We have tabulated the number of motifs that are suited to make accurate predictions along with the total number of extracted motifs (numbers in brackets) in Table II. Given that each motif requires at least $5 \times s$ symbols (s for the motif and $4 \times s$ for the forecast), the local motif repository can be expected to represent the main contributor to PowerSAX's memory requirement on embedded power metering systems. In fact, even when each symbol can be represented in one byte of memory, more than 91 kB of memory are required to store the 5,827 motifs resulting from $w = 64$ and $s = 16$. If the embedded system shall also be capable of emitting the predictions itself instead of just relaying the motif's identifier to the home gateway, this memory demand increases to 455 kB. With current embedded systems in smart metering equipment usually featuring significantly less resources[1], this memory demand immediately renders the presented load forecasting solution inapplicable for such embedded systems.

[1]For example, the Maxim `71M6542F` device designed for its use in smart meters offers 64 kB of program memory and 5 kB of RAM, according to http://www.maximintegrated.com/datasheet/index.mvp/id/6866

C. Operation on Embedded Sensing Systems

In order to complement our analysis of the prediction accuracy, we also assess the resource requirement when running PowerSAX on embedded sensing systems. We have hence set up the evaluation systems based on a typical smart home setup, where a computationally powerful home gateway is commonly present. The home gateway is used for the system's initial configuration. First, it extracts the symbol mappings based on the histogram of the previously observed consumption data. Subsequently, it converts all historic input data to their symbolic representations and extracts the characteristic motifs as well as annotating them by their predictions. Finally, the resulting alphabet and the motifs that have resulted in good predictions are uploaded to the embedded power metering system. We have used the TelosB platform to this end, because it is comparable to the specifications of embedded systems in state-of-the-art smart plugs and smart meters.

We base our evaluation on a simple TinyOS application that takes an analog reading once per second. By default, the application uses 15,642 bytes in ROM and 466 bytes in RAM when compiled for the TelosB. When adding the PowerSAX alphabet as an array that only contains the upper values β_c of each cluster (cf. Eq. (1)), only 2 additional bytes in the program memory are required for each symbol in the alphabet. Our implemented function to convert raw input data to their symbolic representations requires 6 bytes of ROM and no additional RAM.

As highlighted before, storing the motifs has a significantly higher memory demand and thus represents the main limitation when deploying PowerSAX on embedded systems. When implemented in the application introduced above, two new buffers need to be allocated; one of size $\frac{w}{s}$ in order to allow for the averaging over the raw data prior to their conversion to the corresponding symbol, and one of size s, on which the motif recognition is performed. These buffers require between 24 bytes (for $w = 64$ and symbolic lengths of 8 or 16) and 514 bytes (for $w = 512$ and $s = 2$) of RAM. In terms of the program memory, the main contributor is the array of motifs to detect in the real-time data, and s bytes are required for each motif. Hence, between 418 (for $s = 16$) and 3,348 (for $s = 2$) motifs including their predictions can be stored if no further program memory is required by the application. However, when keeping the actual predictions on the home gateway and transferring the motif identifiers instead, the memory available on a TelosB allows for the storage of between 2,092 (for $s = 16$) and over 16,700 (for $s = 2$) motifs on the nodes. The actual implementation to recognize motifs and return their identifier to the home gateway is also very small; it can be implemented in 18 bytes of application code.

While proving the lightweight nature of PowerSAX, our evaluation has shown that the storage of motifs dominates its memory requirement. Consequently, additional assessment steps should be employed on the home gateway in order to reduce the number of used motifs, e.g., by raising the required number of occurrences to add a motif to the repository.

Preprint of: **A. Reinhardt** and S. Koessler. "PowerSAX: Fast Motif Matching in Distributed Power Meter Data Using Symbolic Representations." In: *Proceedings of the 9th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*. 2014, pp. 531–538.

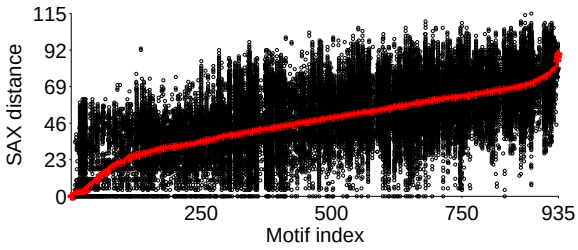


Fig. 6. Prediction accuracy of the original SAX implementation.

D. Comparison to Original SAX

In a final experiment, we compare PowerSAX to the original SAX approach. In both cases, we have set the window size to 512 values of input data, leading to a prediction horizon of 34 minutes for a sampling period of 1 Hz. We have used a symbolic length of $s = 8$. As PowerSAX has extracted nine symbols for these parameters, we have also defined an alphabet of nine symbols for SAX. As SAX applies z-normalization to each sampling window, the resulting prediction errors cannot be measured in watts, but are based on the following different distance definition. For identical and neighboring symbols, the symbolic distance is assumed to be zero (i.e., $dist(C, C) = 0$ and $dist(C, D) = 0$). For all larger distances, such as $dist(B, E)$, the distance is calculated as the difference between their closest cluster boundaries (the largest value of B and the smallest value of E). The results for the analysis of SAX's ability to predict future power consumption are visualized in Fig. 6. A total of 935 distinct motifs could be extracted from the input data, as opposed to 788 motifs for the same configuration of PowerSAX. When allowing for no symbolic distance between the SAX-based prediction and the actual consumption, however, only 36 of 464,229 motif matches, i.e., 0.008%, led to correct predictions, as compared to 11.1% achieved by PowerSAX (cf. Fig. 5).

VII. Conclusion

Consumption data from distributed power meters are a rich resource, upon which manifold novel smart home services can be based. However, the centralized processing of power data streams poses significant requirements to both bandwidth and processing power. To alleviate these problems, we have presented PowerSAX. It converts time series input data to their symbolic representations and allows for the extraction of motifs from these symbolic representations as well as their efficient recognition in real-time data. We have evaluated PowerSAX using real-world power consumption data for the use case of predicting an appliance's future power consumption. The comprehensive analysis of the parameter space has shown that an appliance's future power consumption can be correctly predicted in 96% of the cases ($w = 64$, $s = 16$) when previously extracted motifs are recognized. PowerSAX is lightweight and can be easily integrated into the applications on embedded power metering systems. It thus represents an ideal substrate for the realization of novel smart home automation services based on power consumption data analysis.

VIII. Acknowledgments

We would like to thank Frank Englert for his support.

References

[1] N. Kimura and S. Latifi, "A Survey on Data Compression in Wireless Sensor Networks," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)*, vol. 2, 2005, pp. 8–13.

[2] K. Römer, "Discovery of Frequent Distributed Event Patterns in Sensor Networks," in *Wireless Sensor Networks*, ser. Lecture Notes in Computer Science, R. Verdone, Ed. Springer Berlin Heidelberg, 2008, vol. 4913, pp. 106–124.

[3] S. Zöller, A. Reinhardt, S. Schulte, and R. Steinmetz, "Scoresheet-based Event Relevance Determination for Energy Efficiency in Wireless Sensor Networks," in *Proceedings of the 36th IEEE Conference on Local Computer Networks (LCN)*, 2011, pp. 207–210.

[4] A. Reinhardt, D. Christin, and S. S. Kanhere, "Predicting the Power Consumption of Electric Appliances through Time Series Pattern Matching," in *Proceedings of the 5th ACM Workshop On Embedded Systems For Energy-Efficient Buildings (BuildSys)*, 2013, pp. 1–2.

[5] E. Keogh, J. Lin, and A. Fu, "HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence," in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, 2005, pp. 226–233.

[6] J. Pelt, "Astronomical Time Series Analysis," University Lecture, Oulu University, 2003.

[7] G. Hart, "Nonintrusive Appliance Load Monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[8] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong, "Power Signature Analysis," *IEEE Power and Energy Magazine*, vol. 1, no. 2, pp. 56–63, 2003.

[9] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd, "At the Flick of a Switch: Detecting and Classifying Unique Electrical Events on the Residential Power Line," in *Ubiquitous Computing*, ser. Lecture Notes in Computer Science, J. Krumm, G. D. Abowd, A. Seneviratne, and T. Strang, Eds. Springer Berlin Heidelberg, 2007, vol. 4717, pp. 271–288.

[10] S. Gupta, M. S. Reynolds, and S. N. Patel, "ElectriSense: Single-Point Sensing using EMI for Electrical Event Detection and Classification in the Home," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (UbiComp)*, 2010, pp. 139–148.

[11] L. Jiang, S. Luo, and J. Li, "An Approach of Household Power Appliance Monitoring Based on Machine Learning," in *Proceedings of the 5th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, 2012, pp. 577–580.

[12] T. K. Wijaya, J. Eberle, and K. Aberer, "Symbolic Representation of Smart Meter Data," in *Workshop Proccedings of the 16th International Conference on Extending Database Technology (EDBT)*, 2013, pp. 242–248.

[13] A. Bondu, M. Boullé, and B. Grossin, "SAXO : An Optimized Data-driven Symbolic Representation of Time Series," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–9.

[14] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, 2003, pp. 2–11.

[15] D. Goldin and P. Kanellakis, "On Similarity Queries for Time-Series Data: Constraint Specification and Implementation," in *Proceedings of the 1st International Conference on the Principles and Practice of Constraint Programming (CP)*, 1995, pp. 137–153.

[16] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz, "On the Accuracy of Appliance Identification Based on Distributed Load Metering Data," in *Proceedings of the 2nd IFIP Conference on Sustainable Internet and ICT for Sustainability (SustainIT)*, 2012, pp. 1–9.

[17] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[18] J. Lin, E. Keogh, S. Lonardi, and P. Patel, "Finding Motifs in Time Series," in *Proceedings of the 2nd Workshop on Temporal Data Mining at the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2002, pp. 53–68.